PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ○ The ACM Digital Library ● The Guide

optimizing array reference checking in java programs

THE GUIDE TO COMPUTING LITERATURE

Feedback Report a problem Satisfaction survey

Terms used
optimizing array reference checking in **java programs**

Found **102,211** of **863,039**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The Digital Library

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Compilers: Implicit java array bounds checking on 64-bit architecture
Chris Bentley, Scott A. Watterson, David K. Lowenthal, Barry Rountree
June 2004 **Proceedings of the 18th annual international conference on Supercomputing**
Full text available: pdf(188.75 KB) Additional Information: full citation, abstract, references, index terms

Interest in using Java for high-performance parallel computing has increased in recent years. One obstacle that has inhibited Java from widespread acceptance in the scientific community is the language requirement that all array accesses must be checked to ensure they are within bounds. In practice, array bounds checking in scientific applications may increase execution time by more than a factor of 2. Previous research has explored optimizations to statically eliminate bounds checks, but the dy ...

Keywords: array-bounds checking, java, virtual memory

2 Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3
Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, references, citings, index terms

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compilers, dynamic compilation, interpreters, just-in-time compilers

3 A framework for optimizing Java using attributes
Patrice Pominville, Feng Qian, Raja Vallée-Rai, Laurie Hendren, Clark Verbrugge
November 2000 **Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(314.37 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a framework for supporting the optimization of Java programs using attributes in Java class files. We show how class file attributes may be used to convey both optimization opportunities and profile information to a variety of Java virtual machines including ahead-of-time compilers and just-in-time compilers.We present our work in the context of Soot, a framework that supports the analysis and transformation of Java bytecode (class files)[21, 25, 26]. We demonstrate the frame ...

4 Elimination of Java array bounds checks in the presence of indirection
Mikel Luján, John R. Gurd, T. L. Freeman, José Miguel
November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**

Full text available: pdf(193.97 KB) Additional Information: full citation, abstract, references, index terms

The Java language specification states that every access to an array needs to be within the bounds of that array; i.e. between 0 and array length 1. Different techniques for different programming languages have been proposed to eliminate explicit bounds checks. Some of these techniques are implemented in off-the-shelf Java Virtual Machines (JVMs). The underlying principle of these techniques is that bounds checks can be removed when a JVM/compiler has enough information to guarantee that a seque ...

Keywords: Java, array bounds check, array indirection

5 Effectiveness of cross-platform optimizations for a java just-in-time compiler
Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11

Full text available: pdf(405.65 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, which is the basis for the latest production version of IBM Java JIT compiler that supports a diversity of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and VLIW architectures. In particular, we focus on the design and evaluation of the cross-platform optimizations that are common across different architectures. We studied the effectiveness of each optimization by selectively disabling ...

Keywords: Java, just-in-time compiler, optimization

6 From flop to megaflops: Java for technical computing
José E. Moreira, Samuel P. Midkiff, Manish Gupta
March 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 22 Issue 2

Full text available: pdf(371.84 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Although there has been some experimentation with Java as a language for numerically intensive computing, there is a perception by many that the language is unsuited for such work because of performance deficiencies. In this article we show how optimizing array bounds checks and null pointer checks creates loop nests on which aggressive optimizations can be used. Applying these optimizations by hand to a simple matrix-multiply test case leads to Java-compliant programs whose performance is ...

Keywords: arrays, compilers, java

7 Characterizing the memory behavior of Java workloads: a structured view and opportunities for optimizations
Yefim Shuf, Mauricio J. Serrano, Manish Gupta, Jaswinder Pal Singh
June 2001 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**, Volume 29 Issue 1
Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings

This paper studies the memory behavior of important Java workloads used in benchmarking Java Virtual Machines (JVMs), based on instrumentation of both application and library code in a state-of-the-art JVM, and provides structured information about these workloads to help guide systems' design. We begin by characterizing the inherent memory behavior of the benchmarks, such as information on the breakup of heap accesses among different categories and on the hotness of references to fields and met ...

8 Proceedings of the Second International Workshop on Persistence and Java
Malcolm Atkinson, Mick Jordan
December 1997 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(1.23 MB) Additional Information: full citation, abstract

These proceedings record the Second International Workshop on Persistence and Java, that was held in Half Moon Bay in the San Francisco Bay Area, in August 1997. The focus of the workshop series is the relationship between the Java platform and longterm storage, such as databases and orthogonal persistence. If future application programmers building large and longlived systems are to be well supported, it is essential that the lessons of existing research into language and persistence combinatio ...

9 First International Workshop on Persistence and Java
Malcolm Atkinson, Mick Jordan
November 1996 Technical Report, Sun Microsystems, Inc.
Full text available: pdf(1.54 MB) Additional Information: full citation, abstract

These proceedings record the First International Workshop on Persistence and Java, which was held in Drymen, Scotland in September 1996. The focus of this workshop was the relationship between the Java languages and long-term data storage, such as databases and orthogonal persistence. There are many approaches being taken, some pragmatic and some guided by design principles. If future application programmers building large and long-lived systems are to be well-supported, it is essential that the ...

10 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

11 Runtime optimizations for a Java DSM implementation
R. Veldema, R. F. H. Hofman, R. A. F. Bhoedjang, H. E. Bal
June 2001 **Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande**

Full text available: pdf(740.71 KB) Additional Information: full citation, abstract, references, citings, index terms

Jackal is a fine-grained distributed shared memory implementation of the Java programming language. Jackal implements Java's memory model and allows multithreaded Java programs to run unmodified on distributed-memory systems.

This paper focuses on Jackal's runtime system, which implements a multiple-writer, home-based consistency protocol. Protocol actions are triggered by software access checks that Jackal's compiler inserts before object and array references. We describe optimizatio ...

12 ABCD: eliminating array bounds checks on demand
Rastislav Bodík, Rajiv Gupta, Vivek Sarkar
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5

Full text available: pdf(306.96 KB) Additional Information: full citation, abstract, references, citings, index terms

To guarantee typesafe execution, Java and other strongly typed languages require bounds checking of array accesses. Because array-bounds checks may raise exceptions, they block code motion of instructions with side effects, thus preventing many useful code optimizations, such as partial redundancy elimination or instruction scheduling of memory operations. Furthermore, because it is not expressible at bytecode level, the elimination of bounds checks can only be performed at run time ...

13 Array regrouping and structure splitting using whole-program reference affinity
Yutao Zhong, Maksim Orlovich, Xipeng Shen, Chen Ding
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available: pdf(202.16 KB) Additional Information: full citation, abstract, references, citings, index terms

While the memory of most machines is organized as a hierarchy, program data are laid out in a uniform address space. This paper defines a model of *reference affinity*, which measures how close a group of data are accessed together in a reference trace. It proves that the model gives a hierarchical partition of program data. At the top is the set of all data with the weakest affinity. At the bottom is each data element with the strongest affinity. Based on the theoretical model, the paper p ...

Keywords: array regrouping, program locality, program transformation, reference affinity, reuse signature, structure splitting, volume distance

14 Source-level global optimizations for fine-grain distributed shared memory systems
R. Veldema, R. F. H. Hofman, R. A. F. Bhoedjang, C. J. H. Jacobs, H. E. Bal
June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming**, Volume 36 Issue 7

Full text available: pdf(112.60 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes and evaluates the use of aggressive static analysis in Jackal, a fine-grain Distributed Shared Memory (DSM) system for Java. Jackal uses an optimizing, source-level compiler rather than the binary rewriting techniques employed by most other fine-grain DSM systems. Source-level analysis makes existing access-check optimizations (e.g., access-check batching) more effective and enables two novel fine-grain DSM optimizations: object-graph aggregatio ...

15 Fast, effective code generation in a just-in-time Java compiler
Ali-Reza Adl-Tabatabai, Michał Cierniak, Guei-Yuan Lueh, Vishesh M. Parikh, James M. Stichnoth
May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation**, Volume 33 Issue 5
Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

A "Just-In-Time" (JIT) Java compiler produces native code from Java byte code instructions during program execution. As such, compilation speed is more important in a Java JIT compiler than in a traditional compiler, requiring optimization algorithms to be lightweight and effective. We present the structure of a Java JIT compiler for the Intel Architecture, describe the lightweight implementation of JIT compiler optimizations (e.g., common subexpression elimination, register allocation, and elim ...

16 Design, implementation, and evaluation of optimizations in a just-in-time compiler
Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Mikio Takeuchi, Takeshi Ogasawara, Toshio Suganuma, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
June 1999 **Proceedings of the ACM 1999 conference on Java Grande**
Full text available: pdf(1.09 MB) Additional Information: full citation, references, citings, index terms

17 High performance computing with the Array package for Java: a case study using data mining
J. E. Moreira, S. P. Midkiff, M. Gupta, R. Lawrence
January 1999 **Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)**
Full text available: pdf(178.36 KB) Additional Information: full citation, references, citings, index terms

18 Transformations for model checking distributed Java programs
Scott D. Stoller, Yanhong A. Liu
May 2001 **Proceedings of the 8th international SPIN workshop on Model checking of software**
Full text available: pdf(108.43 KB) Additional Information: full citation, abstract, references

This paper describes three program transformations that extend the scope of model checkers for Java programs to include distributed programs, *i.e.*, multi-process programs. The transformations combine multiple processes into a single process, replace remote method invocations (RMIs) with local method invocations that simulate RMIs, and replace cryptographic operations with symbolic counterparts.

19 Using shape analysis to reduce finite-state models of concurrent Java programs
James C. Corbett
January 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 1
Full text available: pdf(284.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Finite-state verification (e.g., model checking) provides a powerful means to detect concurrency errors, which are often subtle and difficult to reproduce. Nevertheless, widespread use of this technology by developers is unlikely until tools provide automated support for extracting the required finite-state models directly from program source. Unfortunately, the dynamic features of modern languages such as Java complicate the construction of compact finite-state models for verification. I ...

Keywords: Java, concurrent systems, finite-state verification, model extraction, modeling, shape analysis, state-space reductions

20 Data size optimizations for java programs
C. Scott Ananian, Martin Rinard
June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems**, Volume 38 Issue 7
Full text available: pdf(349.36 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a set of techniques for reducing the memory consumption of object-oriented programs. These techniques include analysis algorithms and optimizations that use the results of these analyses to eliminate fields with constant values, reduce the sizes of fields based on the range of values that can appear in each field, and eliminate fields with common default values or usage patterns. We apply these optimizations both to fields declared by the programmer and to implicit fields in the runti ...

Keywords: bitwidth analysis, embedded systems, field externalization, field packing, size optimizations, static specialization

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next